NVM Express Technical Errata

Errata ID	**009**
Change Date	**2/13/2014**
Affected Spec Ver.	**NVM Express 1.0 and NVM Express 1.1a**
Corrected Spec Ver.	

Submission info

Name	Company	Date
Amber Huffman	Intel	11/19/2013
Dave Landsman	SanDisk	11/19/2013
Kevin Marks	Dell	11/19/2013
Peter Onufryk	PMC	12/4/2013
Ken Okin	HGST	12/4/2013
Matthew Wilcox	Intel	12/4/2013
Jason Gao	WD	12/10/2013
Austin Bolen	Dell	12/17/2013
Irma Flores-Mendoza	Seagate	12/17/2013

The autonomous power state section implies that a non-operational state is exited on an I/O Submission Queue Doorbell write only when entered autonomously. This errata clarifies that an I/O Submission Queue Doorbell write causes an exit from an autonomous power state regardless of how it was entered.

Updated the “Number of Dwords in MPTR” to “Number of Dwords in Metadata Transfer” to be precise that the field in Figure 13 conveys the number of Dwords in the metadata transfer pointed to by MPTR.

Updated uses of “weighted round robin with urgent priority class” to be consistent in the specification.

Removed notes from the Admin opcode table that were not required and where one had become out of date with a feature added in NVMe 1.1.

The Format NVM command is updated to show that the Metadata Settings field is not applicable if there is no metadata. The error to flag incorrect transfer settings for the metadata when there is a size of 0h is also removed.

The use of the Expected Logical Block Reference Tag (EILBRT) for reads was clarified in the end-to-end data protection section.

The erratum clarifies that no user data is returned for an affected namespace after a Format NVM command successfully completes.

An informative section on host software recommendations for dealing with asynchronous events (especially persistent conditions) has been added.

Included simplification for the Data Pointer in NVM Command Set commands.

Various editorial corrections are also made.

Description of the specification technical flaw:

Modify the existing section 8.4.1 as shown below (note 8.4.1 is renumbered 8.4.2):

The controller may support autonomous power state transitions, as indicated in the Identify Controller data structure in Figure 82. Autonomous power state transitions provide a mechanism for the host to configure the device to automatically transition between power states on certain conditions without software intervention.

The entry condition to transition to the Idle Transition Power State is that the controller has been in idle for a continuous period of time exceeding the Idle Time Prior to Transition time specified. The controller is idle when there are no commands outstanding to any I/O Submission Queue. The power state to transition to shall be a non-operational power state (a non-operational power state may autonomously transition to another non-operational power state). Refer to section 8.4.1 for more details. ~~In a non-operational power state, no I/O commands are processed by the controller.~~

~~Following an autonomous power state transition to a non-operational state, the controller shall autonomously transition back to the last operational power state when an I/O Submission Queue Tail Doorbell is written.~~

~~Servicing a memory-mapped I/O (MMIO) or configuration register access may cause the controller power to exceed that advertised by the non-operational power state while the access is being serviced, however, the controller shall logically remain in the non-operational power state. Processing a command submitted to the Admin Submission Queue may also cause the controller power to exceed that advertised by the non-operational power state while the command is processed, however, the controller shall logically remain in the current power state unless there is an explicit power state transition requested by a Set Features command with the Power Management feature identifier. When servicing a register access or an Admin command, the controller shall not exceed the maximum power advertised for the last operational power state.~~

Modify Figure 13 and the paragraph preceding the figure as shown below:

In addition to the fields commonly defined for all Admin and NVM commands, Admin and NVM Vendor Specific commands may support the Number of Dwords in Data Transfer and Number of Dwords in ~~MPTR~~ Metadata Transfer fields. If supported, the command format for the Admin Vendor Specific Command and NVM Vendor Specific Commands are defined in Figure 13. For more details, refer to section 8.7.

Figure 13: Command Format – Admin and NVM Vendor Specific Commands (Optional)

Bytes	Description
63:60	**Command Dword 15 (CDW15):** This field is command specific Dword 15.
59:56	**Command Dword 14 (CDW14):** This field is command specific Dword 14.
55:52	**Command Dword 13 (CDW13):** This field is command specific Dword 13.
51:48	**Command Dword 12 (CDW12):** This field is command specific Dword 12.
47:44	**Number of Dwords in ~~MPTR~~ Metadata Transfer (NDM):** This field indicates the number of Dwords in the metadata transfer.
43:40	**Number of Dwords in Data Transfer (NDT):** This field indicates the number of Dwords in the data transfer.
39:16	Refer to Figure 11 for the definition of these fields if it is an Admin command. Refer to Figure 12 for the definition of these fields if it is an NVM or NVM Vendor Specific command.
15:08	Reserved
07:04	**Namespace Identifier (NSID):** This field indicates the namespace ID that this command applies to. If the namespace ID is not used for the command, then this field shall be cleared to 0h. If a command shall be applied to all namespaces on the device, then this field shall be set to FFFFFFFFh. The behavior of a controller in response to an inactive namespace ID for a vendor specific command is vendor specific. Specifying an invalid namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status Invalid Namespace or Format.
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 10.

Modify the fourth paragraph of section 4.8.2 as shown below:

The lowest strict priority class is the ~~Weighed~~ Weighted Round Robin class. This class consists of the three weighted round robin priority levels (High, Medium, and Low) that share the remaining bandwidth using weighted round robin arbitration. Host software controls the weights for the High, Medium, and Low service classes via Set Features. Round robin is used to arbitrate within multiple Submission Queues assigned to the same weighted round robin level. The number of candidate commands that may start processing from each Submission Queue per round is either the Arbitration Burst setting or the remaining weighted round robin credits, whichever is smaller.

Modify the definition of arbitration burst in section 1.6.2 as shown below:

The maximum number of commands that may be launched at one time from a Submission Queue that is using round robin or weighted round robin with urgent priority class arbitration.

Modify the Arbitration Mechanism Supported field in section 3.1.1 in the Controller Capabilities register as shown below:

<table>
<tr>
<td>18:17</td>
<td>RO</td>
<td>Impl Spec</td>
<td>Arbitration Mechanism Supported (AMS): This field is bit significant and indicates the optional arbitration mechanisms supported by the controller. If a bit is set to ‘1’, then the corresponding arbitration mechanism is supported by the controller. Refer to section Error! Reference source not found. for arbitration details.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>17</td><td>Weighted Round Robin with Urgent Priority Class</td></tr>
<tr><td>18</td><td>Vendor Specific</td></tr>
</table>
The round robin arbitration mechanism is not listed since all controllers shall support this arbitration mechanism.</td>
</tr>
</table>

Modify the Arbitration Mechanism Selected field in section 3.1.5 in the Controller Configuration register as shown below:

<table>
<tr>
<td>13:11</td>
<td>RW</td>
<td>0h</td>
<td>Arbitration Mechanism Selected (AMS): This field selects the arbitration mechanism to be used. This value shall only be changed when EN is cleared to ‘0’. Host software shall only set this field to supported arbitration mechanisms indicated in CAP.AMS. If this field is set to an unsupported value, the behavior is undefined.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Round Robin</td></tr>
<tr><td>001b</td><td>Weighted Round Robin with Urgent Priority Class</td></tr>
<tr><td>010b – 110b</td><td>Reserved</td></tr>
<tr><td>111b</td><td>Vendor Specific</td></tr>
</table>
</td>
</tr>
</table>

Modify the Queue Priority field in Figure 53 as shown below:

<table>
<tr>
<td>02:01</td>
<td>Queue Priority (QPRIO): This field indicates the priority ~~service~~ class to use for commands within this Submission Queue. This field is only used when the weighted round robin with ~~an~~ urgent priority ~~service~~ class is the arbitration mechanism ~~is~~ selected, the field is ignored if weighted round robin with ~~an~~ urgent priority ~~service~~ class is not used. Refer to section Error! Reference source not found..
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Urgent</td></tr>
<tr><td>01b</td><td>High</td></tr>
<tr><td>10b</td><td>Medium</td></tr>
<tr><td>11b</td><td>Low</td></tr>
</table>
</td>
</tr>
</table>

Update Figure 38 as shown below:

Figure 38: Opcodes for Admin Commands

Opcode (07)	Opcode (06:02)	Opcode (01:00)	Opcode[2]	O/M[1]	Namespace Identifier Used[3]	Command
Generic Command	Function	Data Transfer				
0b	000 00b	00b	00h	M	No	Delete I/O Submission Queue
0b	000 00b	01b	01h	M	No	Create I/O Submission Queue
0b	000 00b	10b	02h	M	Yes	Get Log Page
0b	000 01b	00b	04h	M	No	Delete I/O Completion Queue
0b	000 01b	01b	05h	M	No	Create I/O Completion Queue
0b	000 01b	10b	06h	M	Yes	Identify
0b	000 10b	00b	08h	M	No	Abort
0b	000 10b	01b	09h	M	Yes	Set Features
0b	000 10b	10b	0Ah	M	Yes	Get Features
0b	000 11b	00b	0Ch	M	No	Asynchronous Event Request
0b	001 00b	00b	10h	O	No	Firmware Activate
0b	001 00b	01b	11h	O	No	Firmware Image Download
	I/O Command Set Specific					
1b	na	na	80h – BFh	O		I/O Command Set specific
	Vendor Specific					
1b	na	na	C0h – FFh	O		Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. A subset of commands uses the Namespace Identifier field (CDW1.NSID). When not used, the field shall be cleared to 0h. ~~For the Get Features and Set Features command, refer to section 7.7 for details on how the Namespace Identifier is used. For the Identify command, the Namespace Identifier is only used for the Namespace data structure. For the Get Log Page command, a value of FFFFFFFFh is used to specify that the global values should be returned.~~

Modify Figure 39 as shown below:

Figure 39: Opcodes for Admin Commands – NVM Command Set Specific

Opcode (07)	Opcode (06:02)	Opcode (01:00)	Opcode[2]	O/M[1]	Namespace Identifier Used[3]	Command
Generic Command	Function	Data Transfer				
1b	000 00b	00b	80h	O	Yes	Format NVM
1b	000 00b	01b	81h	O	Yes	Security Send
1b	000 00b	10b	82h	O	Yes	Security Receive

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. A subset of commands uses the Namespace Identifier field (CDW1.NSID). When not used, the field shall be cleared to 0h.

Modify byte 26 in Figure 84 as shown below:

26	M	**Formatted LBA Size (FLBAS):** This field indicates the LBA data size & metadata size combination that the namespace has been formatted with. Bits 7:5 are reserved. Bit 4 if set to '1' indicates that the metadata is transferred at the end of the data LBA, creating an extended data LBA. Bit 4 if cleared to '0' indicates that all of the metadata for a command is transferred as a separate contiguous buffer of data. Bit 4 is not applicable when there is no metadata. Bits 3:0 indicates one of the 16 supported combinations indicated in this data structure. This is a 0's based value.

Modify bits 15:00 of Figure 85 as shown below:

15:00	**Metadata Size (MS):** This field indicates the number of metadata bytes provided per LBA based on the LBA Data Size indicated. If there is no metadata supported, then this field shall be cleared to 00h. If metadata is supported, then the ~~The~~ namespace may support the metadata being transferred as part of an extended data LBA or as part of a separate contiguous buffer. If end-to-end data protection is enabled, then the first eight bytes or last eight bytes of the metadata is the protection information.

Modify the Metadata Settings field in Figure 111 as shown below:

04	**Metadata Settings (MS):** This field is set to '1' if the metadata is transferred as part of an extended data LBA. This field is cleared to '0' if the metadata is transferred as part of a separate buffer. The metadata may include protection information, based on the Protection Information (PI) field. If the Metadata Size for the LBA Format selected is 0h, then this field is not applicable.

Modify Figure 112 as shown below:

Figure 112: Format NVM – Command Specific Status Values

Value	Description
Ah	**Invalid Format:** The format specified is invalid. This may be due to various conditions, including: 1) specifying an invalid LBA Format number, or 2) enabling protection information when there is not sufficient metadata per LBA, or ~~3) enabling metadata to be transferred as part of a separate buffer when there is no metadata supported as part of the format selected, or~~ 4) invalid security state (refer to TCG SIIS), etc.

Modify bits 39:24 of Figure 12 as shown below:

<table>
<tr><td>39:24</td><td>Data Pointer (DPTR): This field specifies the data used in the command.

If CDW0[15] is cleared to ‘0’, then the definition of this field is:
<table><tr><td>39:32</td><td>PRP Entry 2 (PRP2): This field contains the second PRP entry for the command or if the data transfer spans more than two memory pages, then this field is a PRP List pointer.</td></tr><tr><td>31:24</td><td>PRP Entry 1 (PRP1): This field contains the first PRP entry for the command or a PRP List pointer depending on the command.</td></tr></table>
If CDW0[15] is set to ‘1’, then the definition of this field is:
<table><tr><td>39:24</td><td>SGL Entry 1 (SGL1): This field contains the first SGL segment for the command. If the SGL segment is a Data Block descriptor, then it describes the entire data transfer. If more than one SGL segment is needed to describe the data transfer, then the first SGL segment is a Segment, or Last Segment descriptor. Refer to section Error! Reference source not found. for the definition of SGL segments and descriptor types.</td></tr></table></td></tr>
</table>

Modify Figure 125 as shown below:

Figure 1: Compare – Data Pointer ~~PRP Entries or SGL Entry 1~~

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>127:00</td><td>Data Pointer (DPTR): This field specifies the data to use for the compare. Refer to Figure 12 for the definition of this field.

~~If CDW0[15] is cleared to ‘0’, then the definition of this field is:~~
<table><tr><td>~~127:64~~</td><td>~~PRP Entry 2 (PRP2): This field contains the second PRP entry. If the data to compare is satisfied with PRP Entry 1, then this field is reserved. If the data to compare may be satisfied with two PRP entries total, then this entry specifies the location that should be used for comparison. If the data transfer requires more than two PRP entries, then this field includes a pointer to a PRP List.~~</td></tr><tr><td>~~63:00~~</td><td>~~PRP Entry 1 (PRP1): This field contains the first PRP entry, indicating the location of data that should be used for comparison.~~</td></tr></table>
~~If CDW0[15] is set to ‘1’, then the definition of this field is:~~
<table><tr><td>~~127:00~~</td><td>~~SGL Entry 1 (SGL1): This field contains the first SGL segment for the command, indicating the location of data that should be used for comparison.~~</td></tr></table></td></tr>
</table>

Modify Figure 131 as shown below:

Figure 131: Dataset Management – Data Pointer ~~PRP Entries or SGL Entry 1~~

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>127:00</td><td>
Data Pointer (DPTR): This field specifies the data to use for the command. Refer to Figure 12 for the definition of this field.

~~If CDW0[15] is cleared to '0', then the definition of this field is:~~

<table>
<tr><td>~~127:64~~</td><td>~~PRP Entry 2 (PRP2): This field contains the second PRP entry that specifies the location where data should be transferred from (if there is a physical discontinuity). This field shall not be a pointer to a PRP List.~~</td></tr>
<tr><td>~~63:00~~</td><td>~~PRP Entry 1 (PRP1): Indicates a data buffer that contains the LBA range information. The buffer shall not have more than one physical discontinuity.~~</td></tr>
</table>

~~If CDW0[15] is set to '1', then the definition of this field is:~~

<table>
<tr><td>~~127:00~~</td><td>~~SGL Entry 1 (SGL1): This field contains the first SGL segment for the command, indicating the location of a data buffer that contains the LBA range information.~~</td></tr>
</table>
</td></tr>
</table>

Modify Figure 138 as shown below:

Figure 138: Read – Data Pointer ~~PRP Entries or SGL Entry 1~~

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>127:00</td><td>
Data Pointer (DPTR): This field specifies where data should be transferred to. Refer to Figure 12 for the definition of this field.

~~If CDW0[15] is cleared to '0', then the definition of this field is:~~

<table>
<tr><td>~~127:64~~</td><td>~~PRP Entry 2 (PRP2): This field contains the second PRP entry. If the data transfer is satisfied with PRP Entry 1, then this field is reserved. If the data transfer may be satisfied with two PRP entries total, then this entry specifies the location where data should be transferred to. If the data transfer requires more than two PRP entries, then this field includes a pointer to a PRP List.~~</td></tr>
<tr><td>~~63:00~~</td><td>~~PRP Entry 1 (PRP1): This field contains the first PRP entry, indicating the location where data should be transferred to.~~</td></tr>
</table>

~~If CDW0[15] is set to '1', then the definition of this field is:~~

<table>
<tr><td>~~127:00~~</td><td>~~SGL Entry 1 (SGL1): This field contains the first SGL segment for the command, indicating the location where data should be transferred to.~~</td></tr>
</table>
</td></tr>
</table>

Modify Figure 145 as shown below:

Figure 145: Reservation Acquire – Data Pointer ~~PRP Entries or SGL Entry 1~~

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>127:00</td><td>Data Pointer (DPTR): This field specifies the location of a data buffer where data is transferred from. Refer to Figure 12 for the definition of this field.

~~If CDW0[15] is cleared to '0', then the definition of this field is:~~

<table>
<tr><td>~~127:64~~</td><td>~~PRP Entry 2 (PRP2): This field contains the second PRP entry that specifies the location where data is transferred from (if there is a physical discontinuity). This field shall not be a pointer to a PRP List.~~</td></tr>
<tr><td>~~63:00~~</td><td>~~PRP Entry 1 (PRP1): Indicates a data buffer where data is transferred from.~~</td></tr>
</table>

~~If CDW0[15] is set to '1', then the definition of this field is:~~

<table>
<tr><td>~~127:00~~</td><td>~~SGL Entry 1 (SGL1): This field contains the first SGL segment for the command, indicating the location of a data buffer where data is transferred from.~~</td></tr>
</table>
.</td></tr>
</table>

Modify Figure 149 as shown below:

Figure 149: Reservation Register – Data Pointer ~~PRP Entries or SGL Entry 1~~

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>127:00</td><td>Data Pointer (DPTR): This field specifies the location of a data buffer where data is transferred from. Refer to Figure 12 for the definition of this field.

~~If CDW0[15] is cleared to '0', then the definition of this field is:~~

<table>
<tr><td>~~127:64~~</td><td>~~PRP Entry 2 (PRP2): This field contains the second PRP entry that specifies the location where data is transferred from (if there is a physical discontinuity). This field shall not be a pointer to a PRP List.~~</td></tr>
<tr><td>~~63:00~~</td><td>~~PRP Entry 1 (PRP1): Indicates a data buffer where data is transferred from.~~</td></tr>
</table>

~~If CDW0[15] is set to '1', then the definition of this field is:~~

<table>
<tr><td>~~127:00~~</td><td>~~SGL Entry 1 (SGL1): This field contains the first SGL segment for the command, indicating the location of a data buffer where data is transferred from.~~</td></tr>
</table>
~~.~~</td></tr>
</table>

Modify Figure 152 as shown below:

Figure 152: Reservation Release – Data Pointer ~~PRP Entries or SGL Entry 1~~

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 12 for the definition of this field. ~~If CDW0[15] is cleared to '0', then the definition of this field is:~~ ~~127:64 — **PRP Entry 2 (PRP2):** This field contains the second PRP entry that specifies the location where data is transferred from (if there is a physical discontinuity). This field shall not be a pointer to a PRP List.~~ ~~63:00 — **PRP Entry 1 (PRP1):** Indicates a data buffer where data is transferred from.~~ ~~If CDW0[15] is set to '1', then the definition of this field is:~~ ~~127:00 — **SGL Entry 1 (SGL1):** This field contains the first SGL segment for the command, indicating the location of a data buffer where data is transferred from.~~ .

Modify Figure 155 as shown below:

Figure 155: Reservation Report – Data Pointer ~~PRP Entries or SGL Entry 1~~

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred to. Refer to Figure 12 for the definition of this field. ~~If CDW0[15] is cleared to '0', then the definition of this field is:~~ ~~127:64 — **PRP Entry 2 (PRP2):** This field contains the second PRP entry. If the number of Dwords of the Reservation Status data structure that are to be transferred is satisfield with PRP Entry 1, then this field is reserved. If it is satisfied with two PRP entries total, then this entry specifies the location where data is transferred to. If it requires more than two PRP entries, then this field contains a pointer to a PRP List.~~ ~~63:00 — **PRP Entry 1 (PRP1):** Indicates a data buffer where data is transferred to.~~ ~~If CDW0[15] is set to '1', then the definition of this field is:~~ ~~127:00 — **SGL Entry 1 (SGL1):** This field contains the first SGL segment for the command, indicating the location of a data buffer where data is transferred to.~~ .

Modify Figure 160 as shown below:

Figure 160: Write – Data Pointer ~~PRP Entries or SGL Entry 1~~

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>127:00</td><td>
Data Pointer (DPTR): This field specifies the location of a data buffer where data is transferred from. Refer to Figure 12 for the definition of this field.

~~If CDW0[15] is cleared to '0', then the definition of this field is:~~

<table>
<tr><td>~~127:64~~</td><td>~~PRP Entry 2 (PRP2): This field contains the second PRP entry. If the data transfer is satisfied with PRP Entry 1, then this field is reserved. If the data transfer may be satisfied with two PRP entries total, then this entry specifies the location where data should be transferred from. If the data transfer requires more than two PRP entries, then this field includes a pointer to a PRP List.~~</td></tr>
<tr><td>~~63:00~~</td><td>~~PRP Entry 1 (PRP1): This field contains the first PRP entry, indicating the location of data that should be transferred from.~~</td></tr>
</table>

~~If CDW0[15] is set to '1', then the definition of this field is:~~

<table>
<tr><td>~~127:00~~</td><td>~~SGL Entry 1 (SGL1): This field contains the first SGL segment for the command, indicating the location of data that should be transferred from.~~</td></tr>
</table>
</td></tr>
</table>

Update the second paragraph after Figure 181 in section 8.3 as follows:

Checking of protection information consists of the following operations performed by the controller. If bit 2 of the Protection Information Check (PRCHK) field of the command is set to '1', then the controller compares the protection information Guard field to the CRC-16 computed over the logical block data. If bit 1 of the PRCHK field is set to '1', then the controller compares unmasked bits in the protection information Application Tag field to the Logical Block Application Tag (LBAT) field in the command. A bit in the protection information Application Tag field is masked if the corresponding bit is cleared to '0' in the Logical Block Application Tag Mask (LBATM) field of the command. If bit 0 of the PRCHK field is set to '1', then the controller compares the protection information Reference Tag field to the computed reference tag. The value of the computed reference tag for the first LBA of the command is the value contained in the Initial Logical Block Reference Tag (ILBRT) or Expected Initial Logical Block Reference Tag (EILBRT) field in the command, for writes and reads respectively. The computed reference tag is incremented for each subsequent logical block. Unlike DIF Type 1 protection which implicitly uses the least significant four bytes of the LBA, The controller always uses the ILBRT or EILBRT field and requires host software to initialize the ILBRT or EILBRT field to the least significant four bytes of the LBA when Type 1 protection is used.

Modify the first paragraph of section 5.13 as shown below:

The Format NVM command is used to low level format the NVM media. This is used when the host wants to change the LBA data size and/or metadata size. A low level format may destroy all data and metadata associated with all namespaces or only the specific namespace associated with the command (refer to the Format NVM Attributes field in the Identify Controller data structure). After the Format NVM command successfully completes, the controller shall not return any user data that was previously contained in an affected namespace.

Add the following new section after section 7.6:

7.7 Asynchronous Event Request Host Software Recommendations (Informative)

This section describes the recommended host software procedure for Asynchronous Event Requests.

The host sends *n* Asynchronous Event Request commands (refer to section 7.6.1, step 11). When an Asynchronous Event Request completes (providing Event Type, Event Information, and Log Page details):

1. Host software issues a Set Features command for the Asynchronous Event Configuration feature specifying to disable reporting all events that utilize the Log Page reported. Host software should wait for the Set Features command to complete.
2. Host software issues a Get Log Page command requesting the Log Page reported as part of the Asynchronous Event Command completion. Host software should wait for the Get Log Page command to complete.
3. Host software parses the returned Log Page. If the condition is not persistent, then host software should re-enable all asynchronous events that utilize the Log Page. If the condition is persistent, then host software should re-enable all asynchronous events that utilize the Log Page except for the one(s) reported in the Log Page. The host re-enables events by issuing a Set Features command for the Asynchronous Event Configuration feature.
4. Host software should issue an Asynchronous Event Request command to the controller (restoring to *n* the number of these commands outstanding).
5. If the condition reported was persistent, host software should continue to monitor the event (e.g., over temperature threshold) to determine if reporting of the event should be re-enabled.

Modify the first paragraph of section 2.6.3 as shown below:

This register controls the reporting of the individual errors by the controller. A masked error is not reported in the ~~in the~~ Header Log register (AERHL), does not updated the First Error Pointer (AERCC.FEP), and is not reported to the host. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Modify the paragraph that precedes Figure 3 as shown below:

Figure 3 shows an NVM subsystem that contains a single NVM Express controller and a single PCI Express port. Since this is a single Function PCI Express device, the NVM Express controller shall be associated with PCI Function 0. A controller may support multiple namespaces. The controller in ~~Figure x1~~ Figure 3 supports two namespaces labeled NS A and NS B. Associated with each controller namespace is a namespace ID, labeled as NSID 1 and NSID 2, that is used by the controller to reference a specific namespace. The namespace ID is distinct from the namespace itself and is the handle a host and controller use to specify a particular namespace in a command. The mapping of a controller's namespace IDs to namespaces is outside the scope of this specification. In this example namespace ID 1 is associated with namespace A and namespace ID 2 is associated with namespace B. Both namespaces are private to the controller and this configuration supports neither multi-path I/O nor namespace sharing.

Modify the title of Figure 4 as shown below:

Figure 4: NVM Subsystem with Two ~~Controller~~ Controllers and One Port

Modify the title of Figure 5 as shown below:

Figure 5: NVM Subsystem with Two ~~Controller~~ Controllers and Two Ports

Modify the paragraph that precedes Figure 5 by inserting a comma in the first sentence, as shown below:

Figure 5 illustrates an NVM Subsystem with two PCI Express ports, each with an associated controller. Both controllers map to PCI Function 0 of the corresponding port. Each PCI Express port in this example is completely independent and has its own PCI Express Fundamental Reset and reference clock input. A reset of a port only affects the controller associated with that port and has no impact on the other controller, shared namespace, or operations performed by the other controller on the shared namespace. The functional behavior of this example is otherwise the same as that illustrated in Figure 4.

Modify the first paragraph of section 1.4.1 as shown below:

This section provides an overview of multi-path I/O and namespace sharing. Multi-path I/O refers to two or more completely independent ~~physical~~ PCI Express paths between a single host and a namespace while namespace sharing refers to the ability for two or more hosts to access a common shared namespace using different NVM Express controllers. Both multi-path I/O and namespace sharing require that the NVM subsystem contain two or more controllers. Concurrent access to a shared namespace by two or more hosts requires some form of coordination between hosts. The procedure used to coordinate these hosts is outside the scope of this specification.

Modify the first paragraph of section 8.8.2 as shown below:

Prior to establishing a reservation on a namespace, a host shall become a registrant of that namespace by registering a reservation key. This reservation key may be used as a means of identifying the registrant (host), authenticating the registrant, and preempting a failed or uncooperative registrant ~~the registrant should the registrant fail or become uncooperative~~. The value of the reservation key used by a host and the method used to select its value is outside the scope of this specification.

Modify Figure 83 as shown below:

25	**Non-Operational State (NOPS):** This field indicates whether the controller processes I/O commands in this power state. If this field is cleared to '0', then the controller processes I/O commands in this power state. If this field is set to '1', then the controller does not process I/O commands in this power state. Refer to section 8.4.1.

Insert this new section 8.4.1 as shown below:

8.4.1 Non-Operational Power States

A power state may be a non-operational power state, as indicated by Non-Operational State (NOPS) field in Figure 83. In a non-operational power state, memory-mapped I/O (MMIO) accesses, configuration register accesses and Admin Queue commands are serviced. No I/O commands are processed by the controller while in a non-operational power state.

When in a non-operational power state, regardless of whether autonomous power state transitions are enabled, the controller shall autonomously transition back to the last operational power state when an I/O Submission Queue Tail Doorbell is written.

Servicing a memory-mapped I/O (MMIO) or configuration register access may cause the controller power to exceed that advertised by the non-operational power state while the access is being serviced, however, the controller shall logically remain in the non-operational power state. Processing a command submitted to the Admin Submission Queue may also cause the controller power to exceed that advertised by the non-operational power state while the command is processed, however, the controller shall logically remain in the current power state unless there is an explicit power state transition requested by a Set Features command with the Power Management feature identifier. When servicing a register access or an Admin command, the controller shall not exceed the maximum power advertised for the last operational power state.

Disposition log

11/19/2013	Erratum captured.
12/4/2013	Added format clarification and asynchronous event recommendations.
12/10/2013	Added overview of a few changes. Fixed Data Pointer example change. Added editorial fix.
12/17/2013	Simplified Data Pointer for the NVM commands and added editorial fixes.
12/20/2013	Editorial update.
2/13/2014	Updated based on feedback from 30-day review to clarify non-operational states are not tied to autonomous transitions.
4/14/2014	Erratum ratified